

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

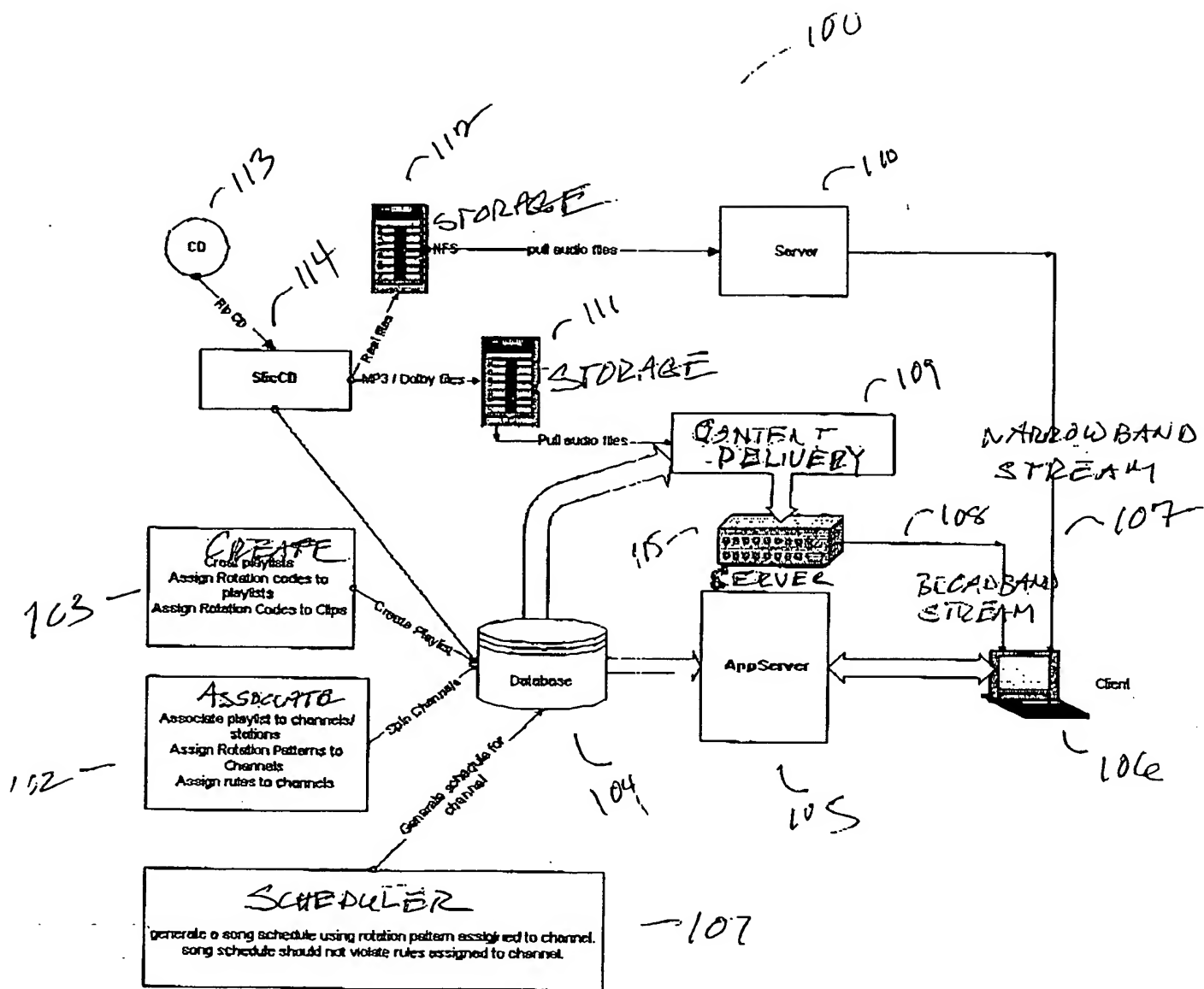


Figure 1

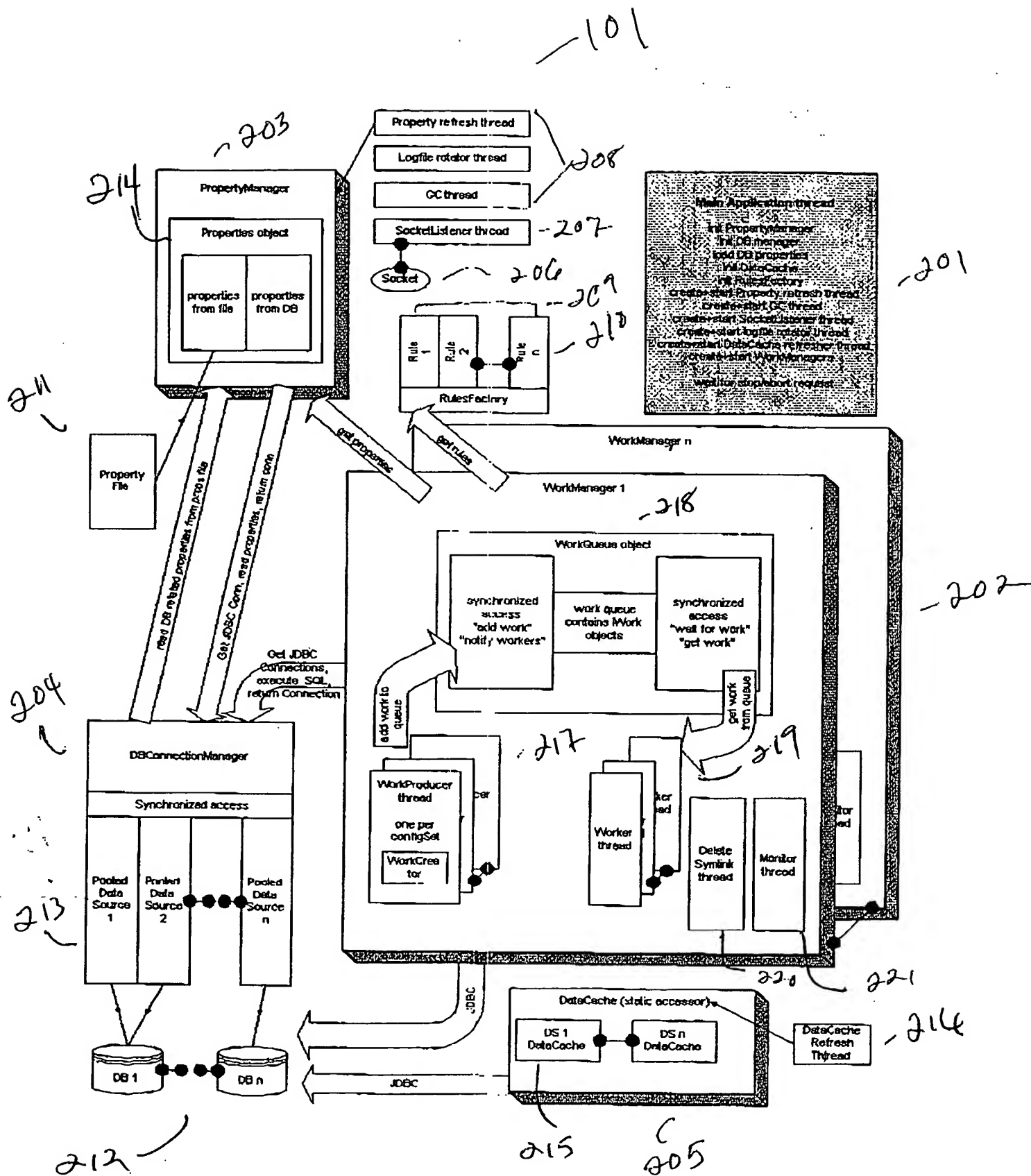


Figure 2

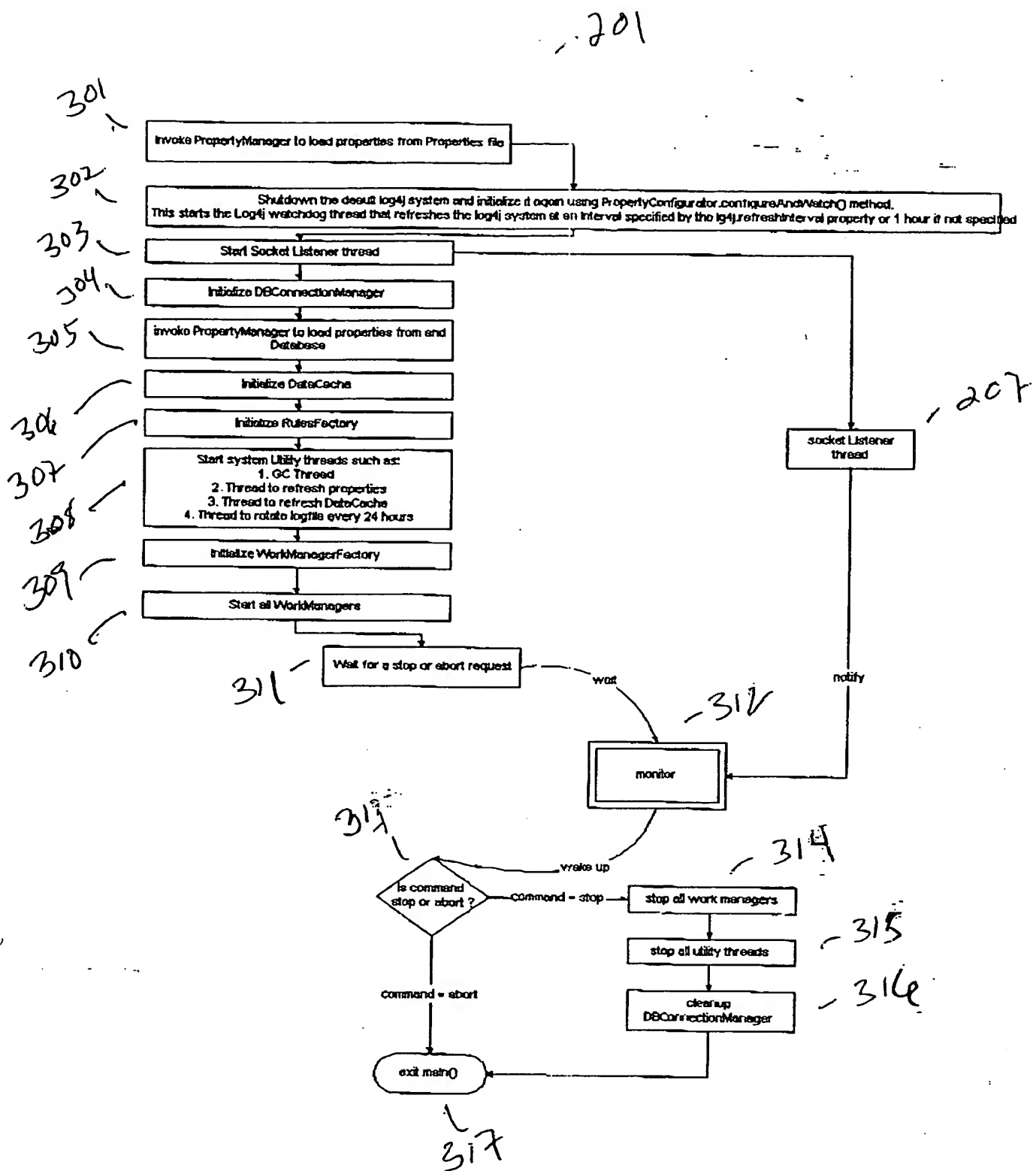


Figure 3

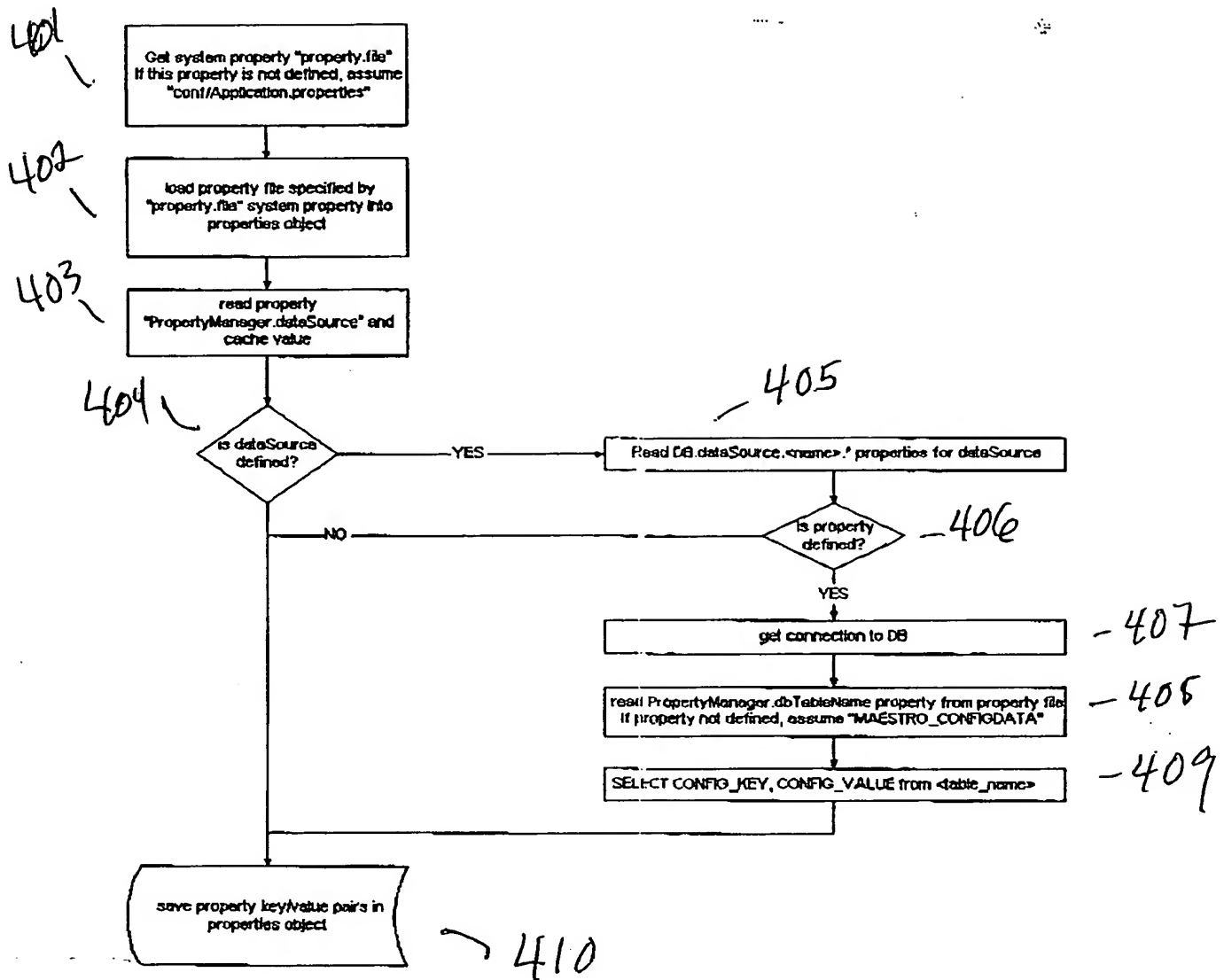


Figure 4

304

501

502

read property "DB.dataSources".
This specifies the number/names of
DataSources to be created.

Create Hashtable to hold
PooledDataSource objects

for each
dataSource name

create a PooledDataSource instance

Register oracle.jdbc.driver.OracleDriver() with the DriverManager

Read following properties for dataSource

- DB.dataSource.<dsName>.URL
- DB.dataSource.<dsName>.username
- DB.dataSource.<dsName>.password
- DB.dataSource.<dsName>.initialPoolSize
- DB.dataSource.<dsName>.maxPoolSize
- DB.dataSource.<dsName>.incrementBy
- DB.dataSource.<dsName>.autoShrink

Open connections to DB specified by above properties.
Num initial connections = initialPoolSize property

Add PooledDataSource object to Hashtable using the dataSourceName as the key

503

504

end of loop

Cache PooledDataSource names and
instances in DBConnectionManager

505

Figure 5

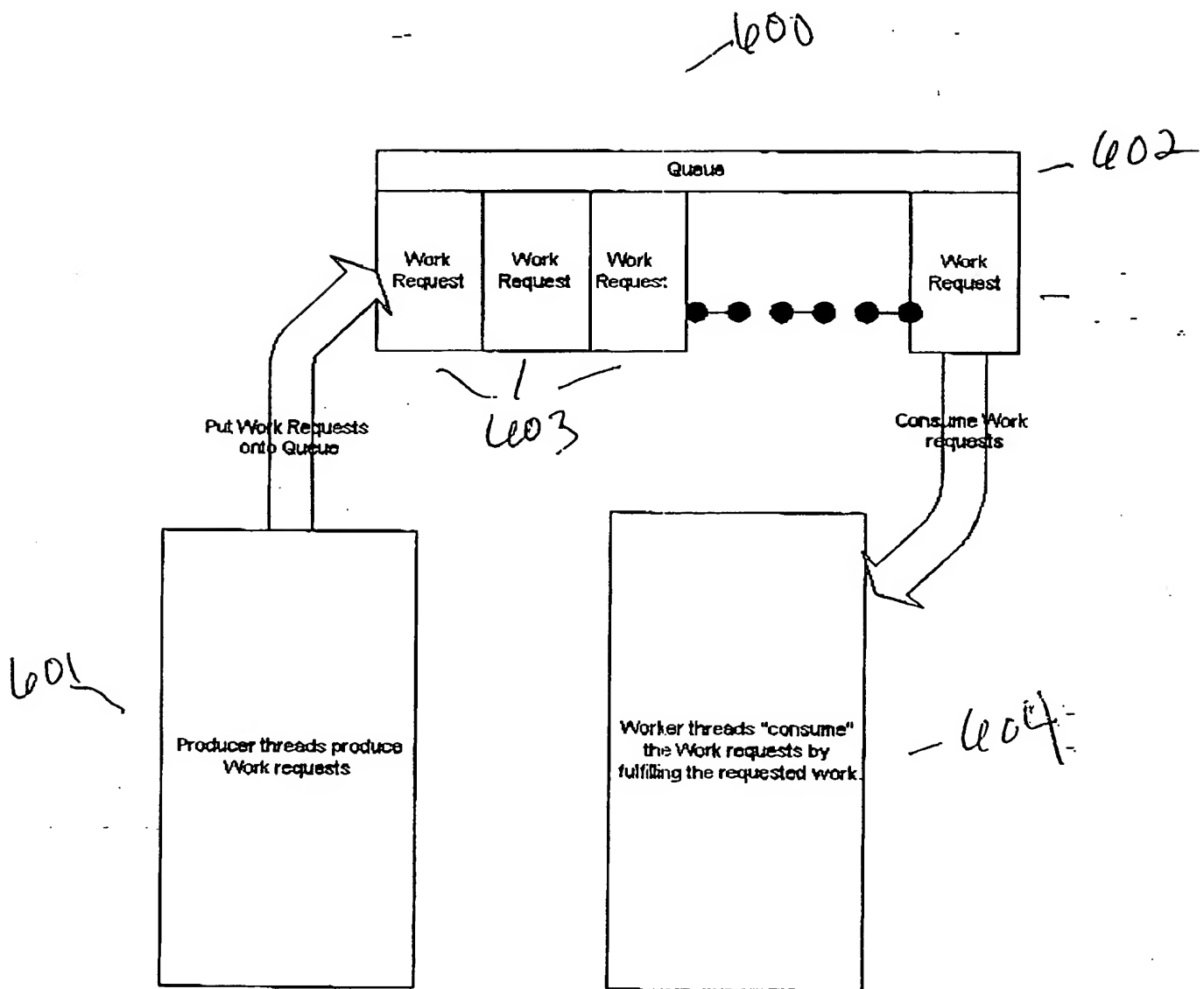


Figure 6

- 509

701-

read property
"WorkManager.instances".
This specifies the number/names of
WorkManagers to be created.

for each work
manager name

create a WorkManager instance

- 702

Create a WorkQueue object

- 703

Create ThreadGroup for Worker threads. Name of threadgroup =
<workmanagename>.WorkerGroup

- 704

read property "WorkManager.<name>.workerThreadPoolInitSize"
and create as many worker threads in the Worker thread group

- 705

Create ThreadGroup for WorkerProducer threads. Name of
threadgroup = <workmanagename>.WorkProducerGroup

- 706

read property "WorkManager.<name>.workCreatorFactories".
This gives a comma separated list of class names.
Create a class object for each class name
These factory classes implement WorkCreatorFactory interface.
Instantiate an Object for each class.
Invoke the getWorkCreators() method on these WorkCreatorFactory objects.
This method returns an array of WorkCreator objects.
For each WorkCreator object, create a WorkProducer thread wrapping the
WorkCreator object.

- 707

read "WorkManager.<name>.monitoringEnabled" property. If set to
true, then create a MonitoringThread

- 708

create a DeleteSymLink thread

- 709

Thus we have a WorkManager object encapsulating the
following:

1. A group of WorkProducer threads
2. A group of Worker threads.
3. A WorkQueue
4. A DeleteSymLink thread
5. An optional MonitoringThread

- 710

end of loop

Cache WorkManager names and
instances in
WorkManagerFactory

711

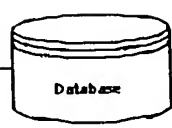
Figure 7

800

801

Get following data for channel:
1. list of sequences from CHANNEL_SEQUENCES
2. Current sequence from MAESTRO_STATUS
3. current index within current sequence from MAESTRO_STATUS
4. AD Interval from CHANNEL table
5. rules to be applied from CHANNEL_RULES table
6. last_ad_scheduled time from SONG_PLAY, if applicable
7. get DAILY_RESET time from CHANNEL or MAESTRO_CONFIGDATA
Create a Random number generator seeded with System.currentTimeMillis and store in Channel object

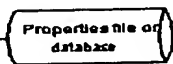
This data is made available via Channel object



212

802

8. Get size of playlist to be maintained (typically 7 hours) let call this "buffer_size". This is set in props file (or: database)



211, 212

803

determine sched_start of next song:
 $next_sched_start = last_sched_start + last_song_length \text{ in milliseconds}$

804

next_scheduled_start > current_time + buffer_size?

YES

log INFO: done

return

NO

num_iterations >= 600
(to prevent infinite loop)

YES

log ERROR: infinite loop.

NO

805

last_scheduled_start < DAILY_RESET_TIME
AND
next_scheduled_start > DAILY_RESET_TIME?

YES

log INFO: doing RESET

reset
Channel current_sequence_index to 0, so that next song scheduled will be based on first code in sequence

NO

806

Schedule ADs for this time

807

NO

scheduleSong

810

is it time for an AD?
This check is based on ad_interval for channel and time last AD was scheduled

808

NO

Channel: last_ad_scheduled

YES

scheduleAD

809

Figure 8

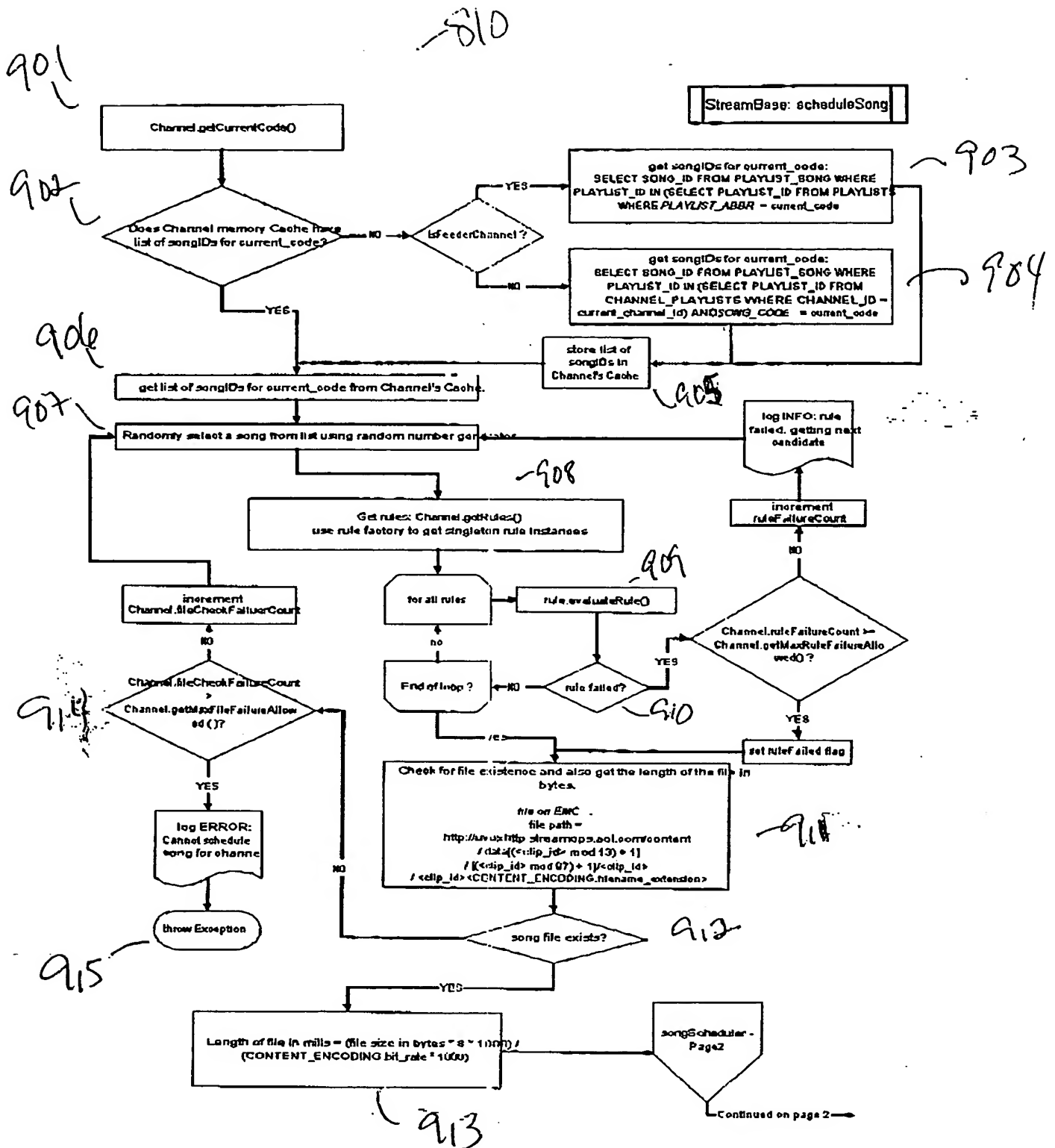


Figure 9

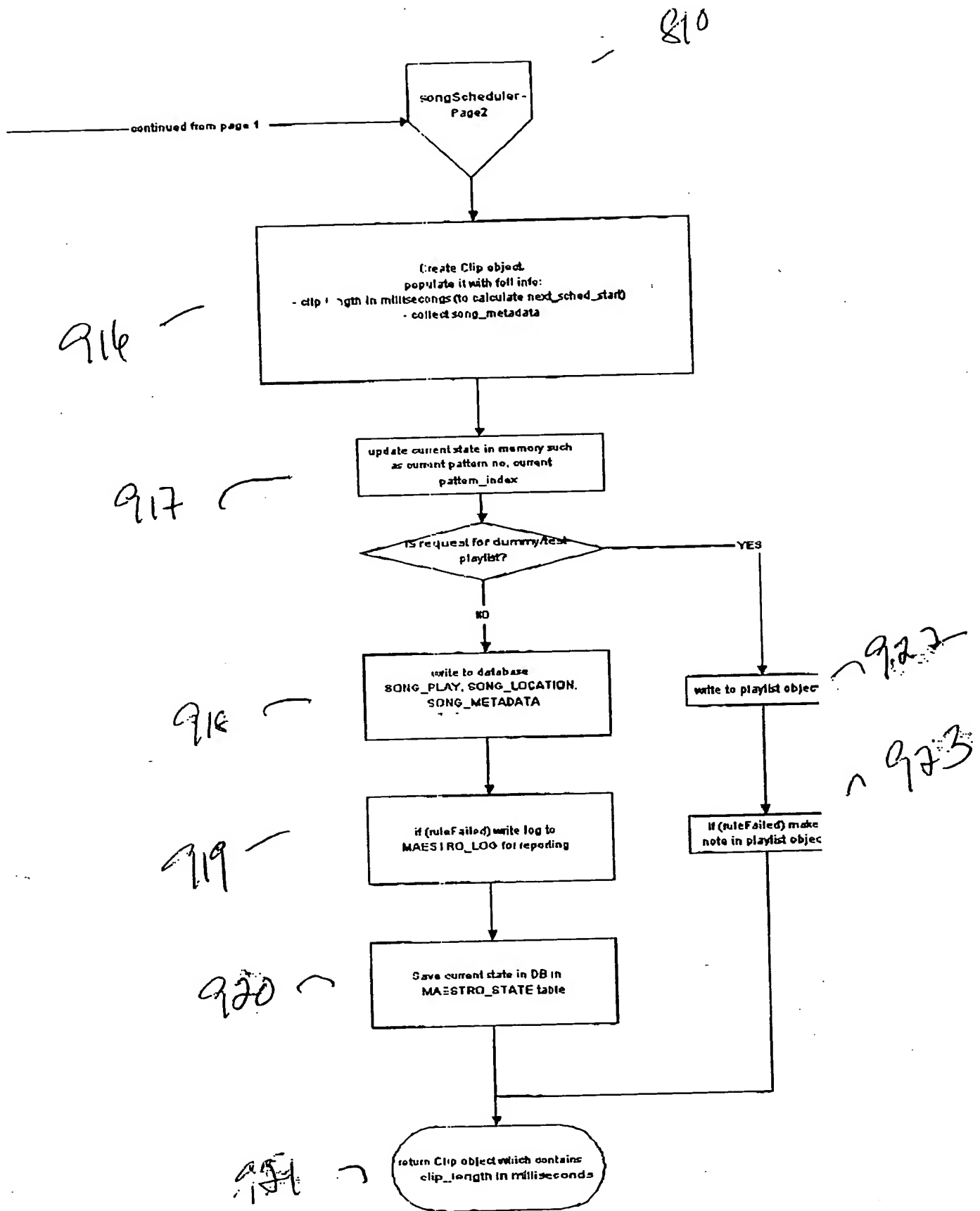
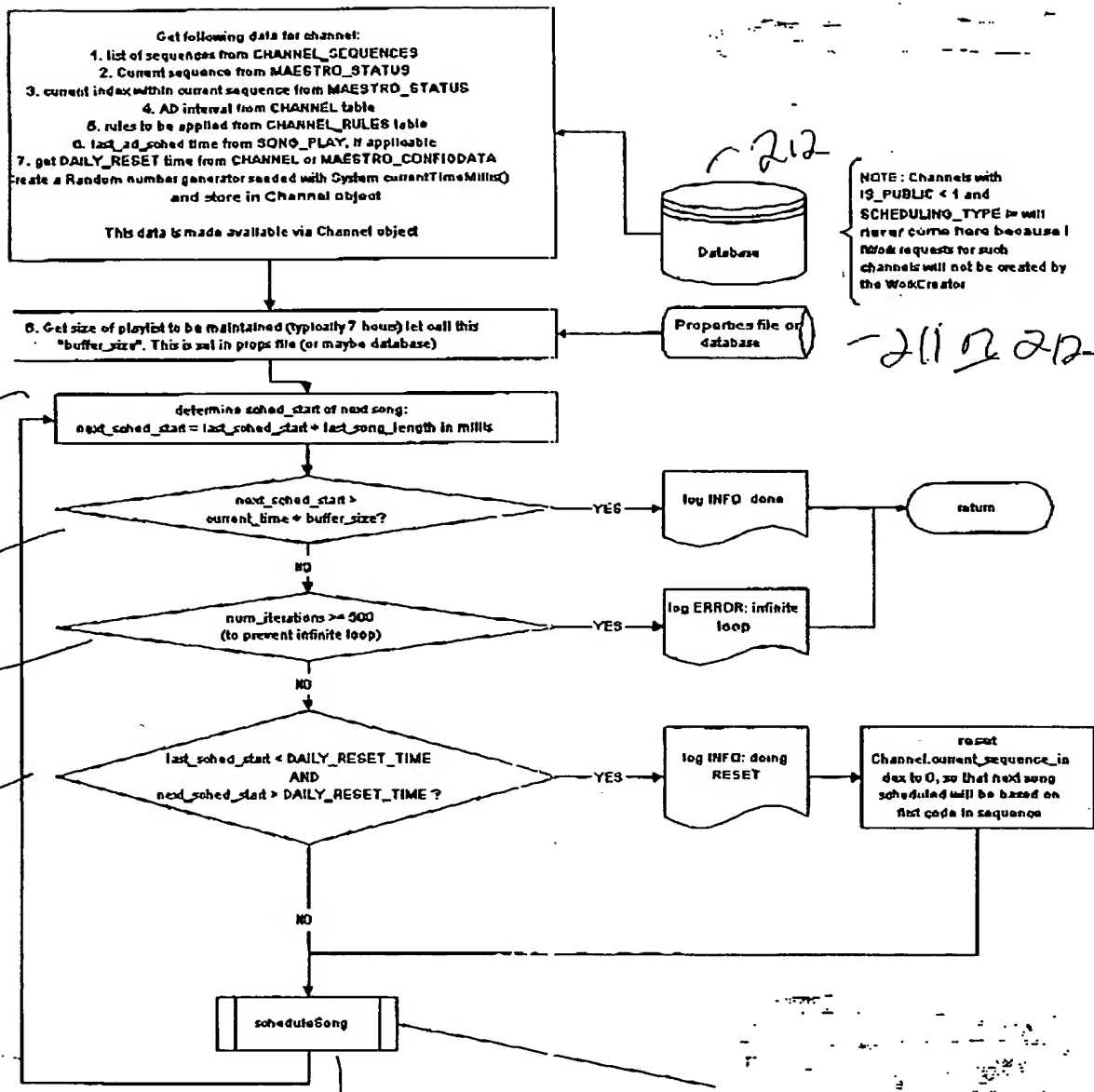


Figure 10

1100



NOTE: Channels with IS_PUBLIC < 1 and SCHEDULING_TYPE != will never come here because I don't see requests for such channels will not be created by the WorkCreator

211 212 212

Note on AD Scheduling : For version 1.0 of Maestro, AD scheduling will be done by a separate thread in the system, not the SongScheduler. When we integrate with LightningCast, AD scheduling will be done as part of the SongScheduling logic.

Figure 11

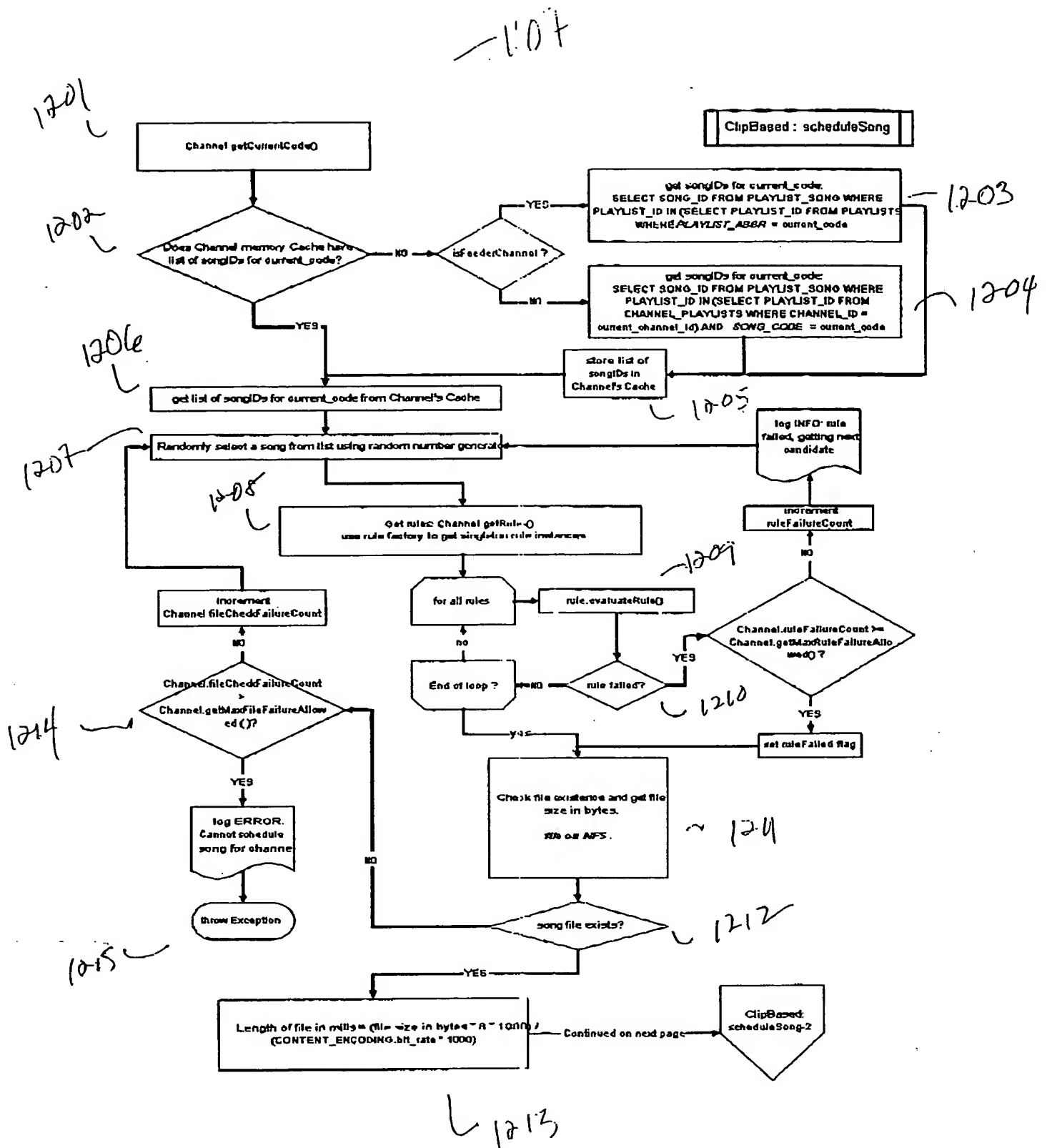


Figure 12

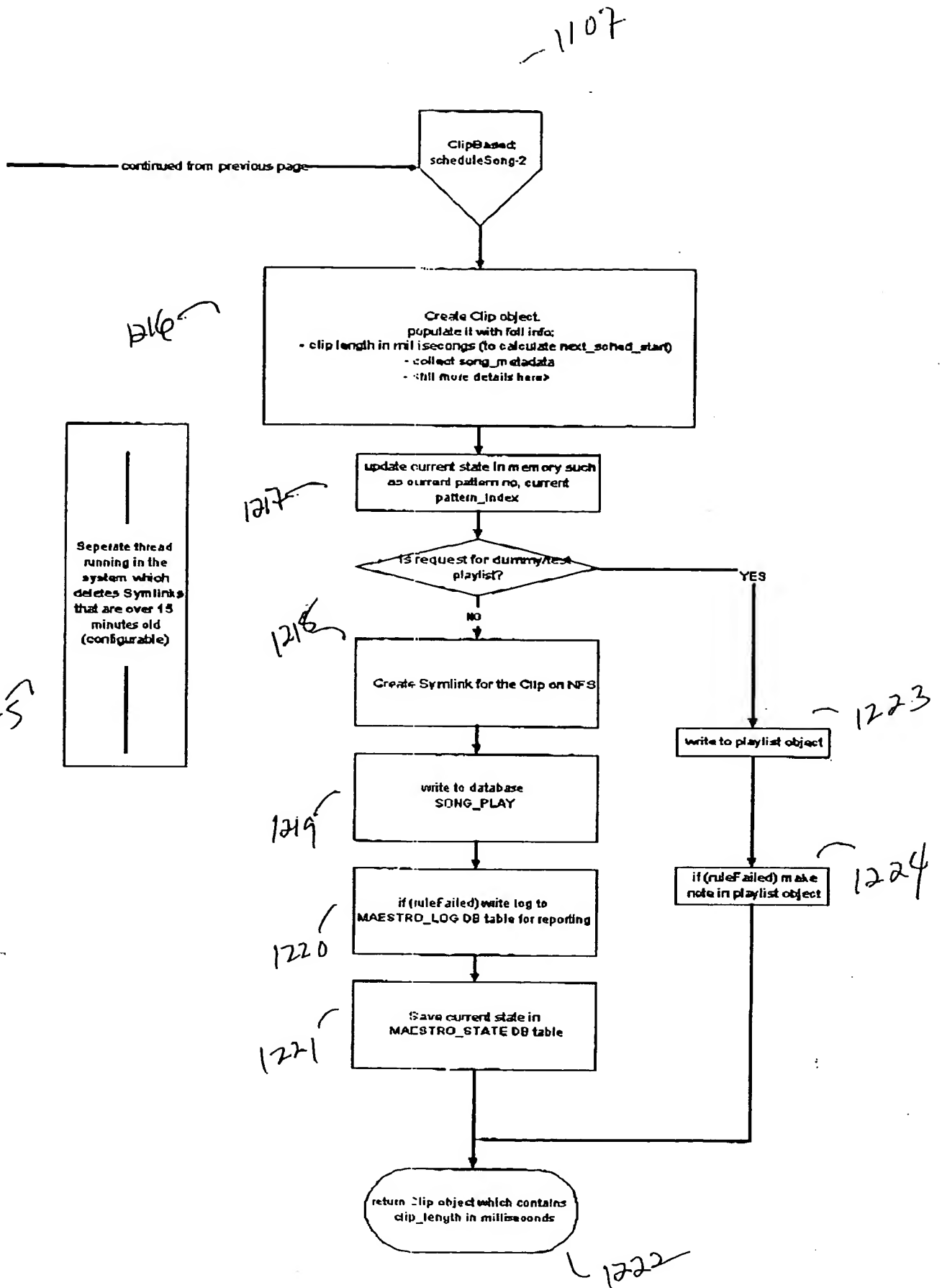


Figure 13

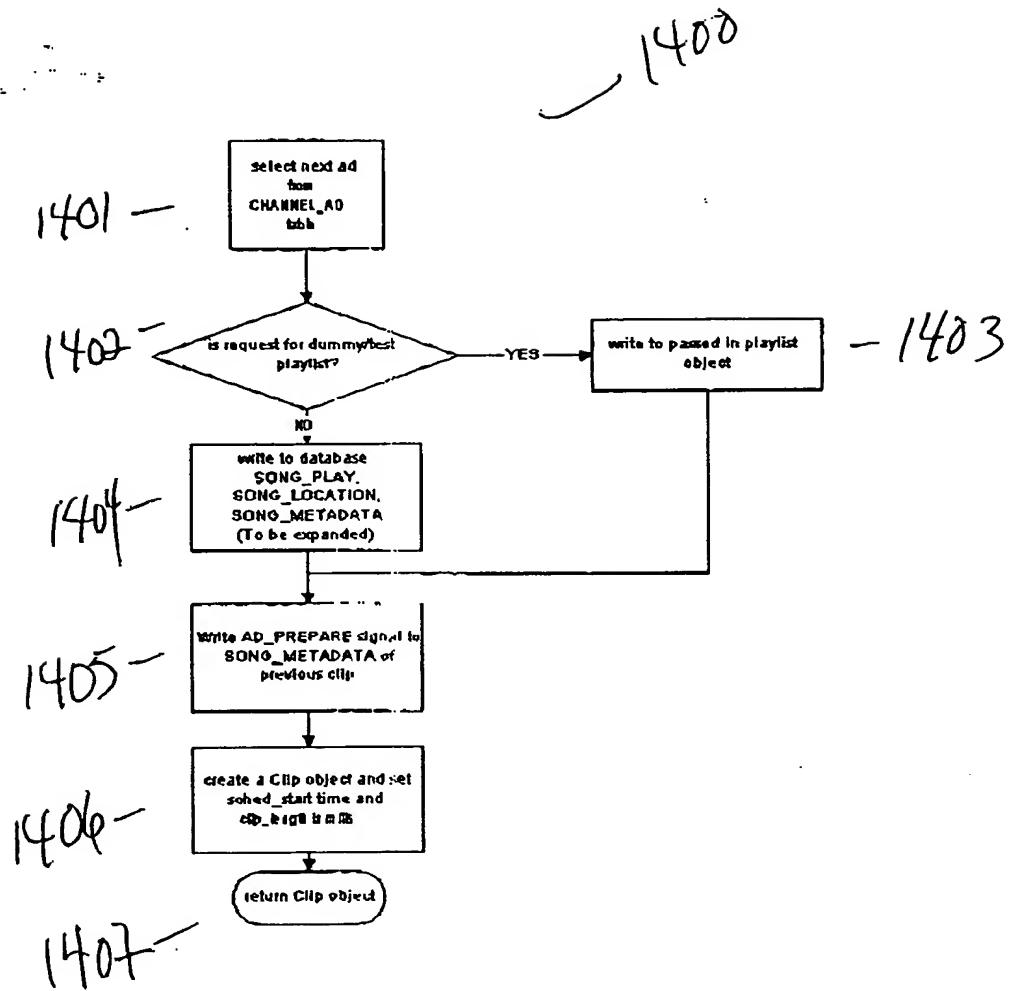


Figure 14

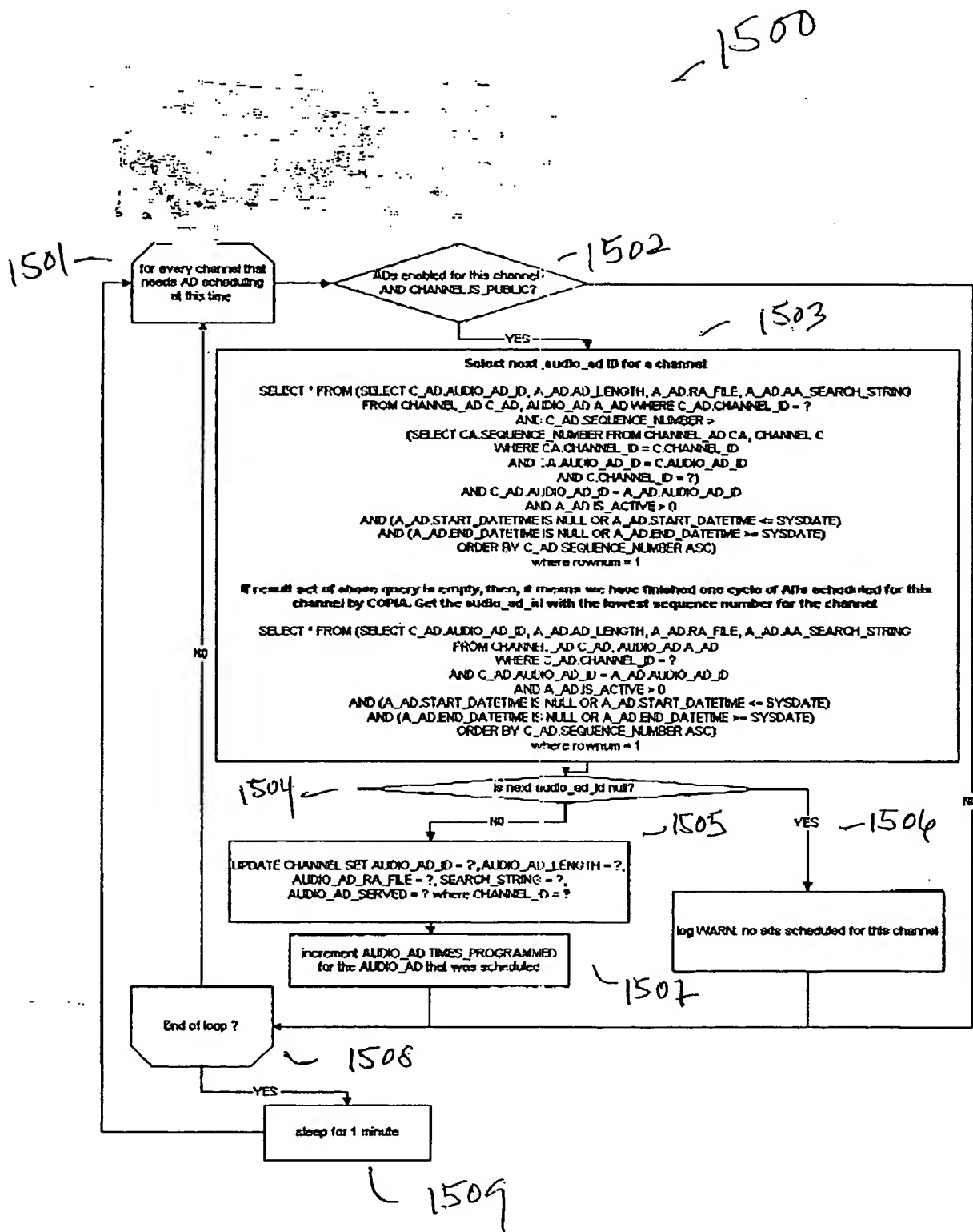


Figure 15

- 307

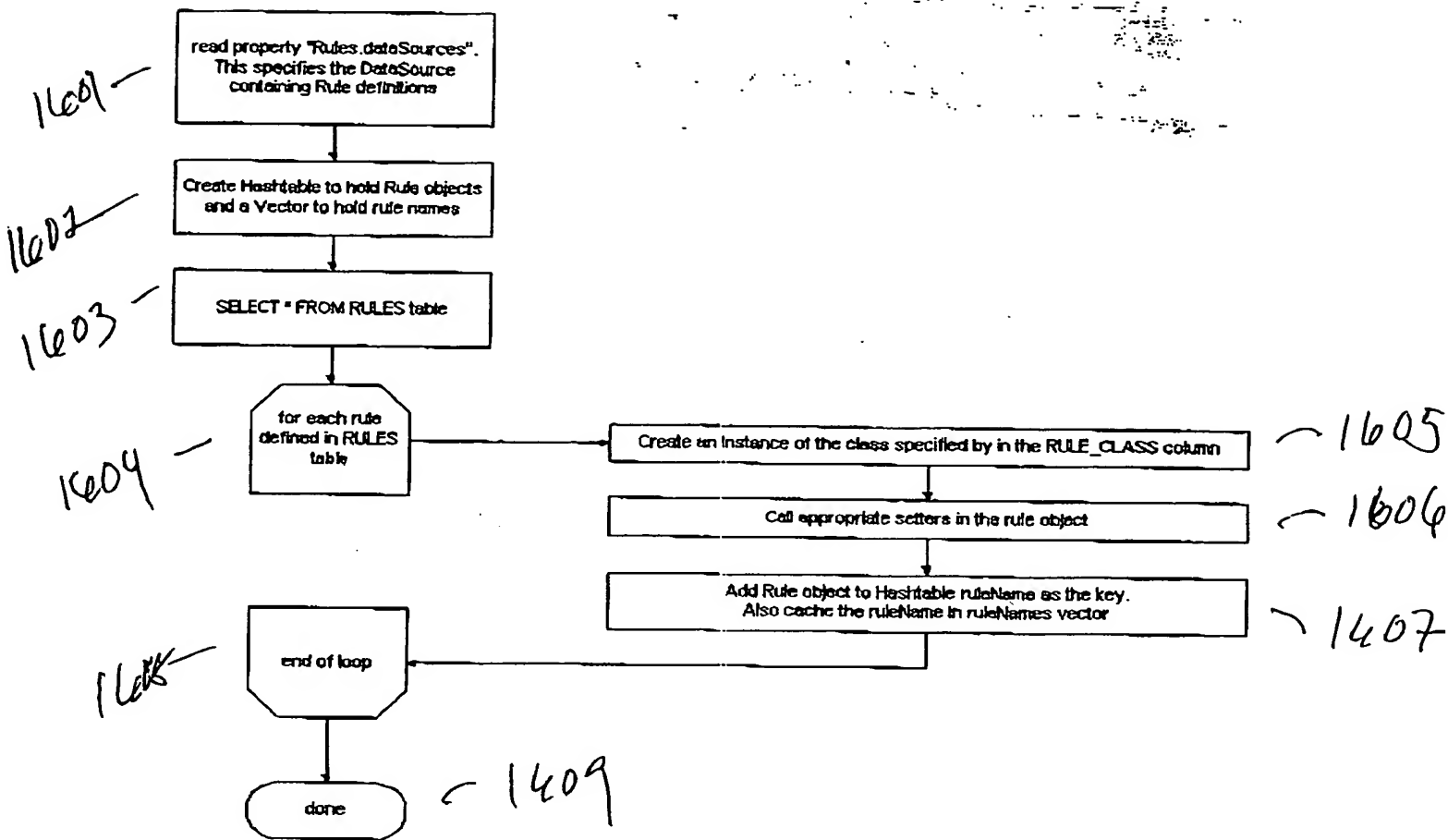


Figure 16

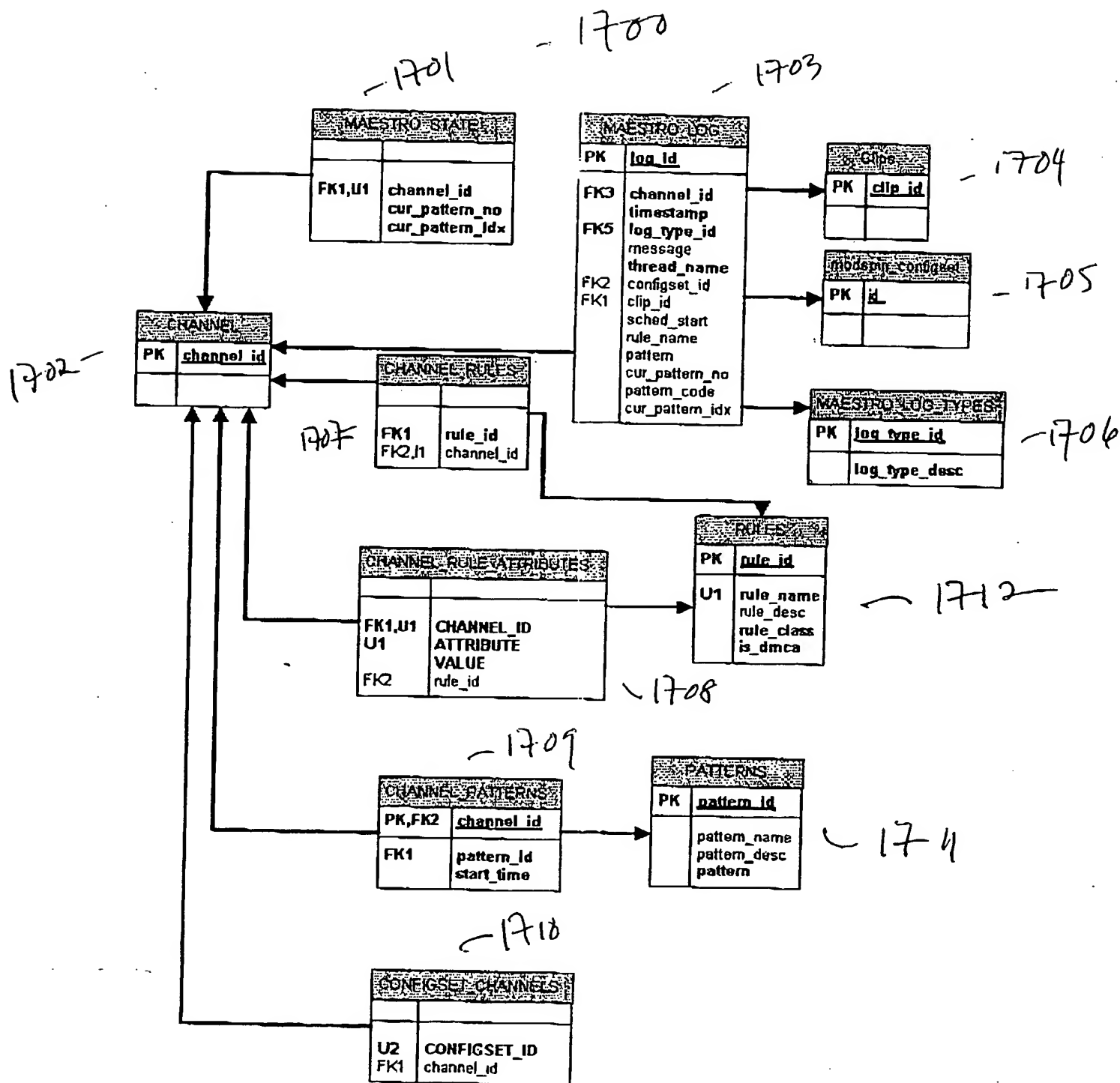


Figure 17